Gaining analytic control of parton showers

Christian W. Bauer and Frank J. Tackmann
Ernest Orlando Lawrence Berkeley National Laboratory, University of California, Berkeley, CA 94720

Parton showers are widely used to generate fully exclusive final states needed to compare theoretical models to experimental observations. While, in general, parton showers give a good description of the experimental data, the precise functional form of the probability distribution underlying the event generation is generally not known. The reason is that realistic parton showers are required to conserve four-momentum at each vertex. In this paper we investigate in detail how four-momentum conservation is enforced in a standard parton shower and why this destroys the analytic control of the probability distribution. We show how to modify a parton shower algorithm such that it conserves four-momentum at each vertex, but for which the full analytic form of the probability distribution is known. We then comment how this analytic control can be used to match matrix element calculations with parton showers, and to estimate effects of power corrections and other uncertainties in parton showers.

I. INTRODUCTION

To analyze experimental data at high energy colliders, one needs precise theoretical predictions to compare measurements against. For such comparisons, Monte Carlo event generators that simulate fully exclusive events are indispensable tools [1]. While it is possible to calculate simple observable distributions analytically, in most cases a direct comparison of such calculations with the data is very difficult, because the experimental analyses have to impose a variety of cuts, and detector efficiencies are, in general, not uniform over the measured distribution.

Event generators typically simulate events in three separate steps: First, a matrix element generator generates an event with few final-state partons, based on full matrix element calculations, which include all interference effects of the quantum field theory. Second, a parton shower adds additional partons using classical splitting probabilities. Finally, all partons are turned into hadrons according to some QCD model of hadronization.

In this paper we are concerned with the second step in an event generator, the parton shower. In practice it is impossible to generate all possible partonic final states using matrix element calculations, because the number of partons in the final state quickly becomes too large. In contrast, parton showers are based on splitting functions which describe the classical probability that a given mother parton splits into two daughter partons. Thus, with a certain probability described by the splitting functions, the parton shower turns a final state containing n partons into one with $n+1$ partons. The advantage of using splitting functions over full matrix elements is that this procedure can be iterated to take a simple final state with a small number of partons and produce many additional partons using a Markov Chain.

The splitting functions describe the splitting in the collinear limit, where the mother and two daughters have large energy and small invariant mass. The parton shower also resums the leading Sudakov double logarithms. Recently it was shown [2] that the parton shower is formally reproduced as the leading order in an effective field theory expansion using soft-collinear effective theory [3], which provides, in principle, a framework for systematic improvements. For example, one could attempt to sum large logarithms beyond the leading order or study power corrections. However, almost all theoretical improvements will necessarily go beyond the level of classical splitting probabilities. It is thus very unlikely that they can be incorporated into a Markov Chain algorithm. A typical example are matrix element calculations, for which it is hard to directly distribute events according to, and which must be explicitly matched with parton showers to avoid double counting [4, 5, 6].

Other important aspects are theoretical uncertainties in the generated distribution [7]. They arise from input parameters, like α_s and quark masses, as well as from higher-order power and perturbative corrections. Having a reliable estimate of these uncertainties becomes crucial at the LHC in searches for New Physics signals where the Standard Model background can only be estimated using Monte Carlo generators.

Related to these issues, important practical considerations must be taken into account as well. The three steps described above produce a theoretical distribution of events. However, the obtained events still have to be run through a detector simulation in order to compare them directly with experimental data. It is this additional step which consumes by far the most computing time in practice. Generating a typical event at the LHC before the detector simulation takes only a fraction of a second per CPU, whereas propagating an event through the detector simulation takes several minutes. In addition, it takes of the order of a megabyte to store a fully simulated event. The available amount of processing power and disk space thus leads to considerable practical limitations. For example, it is extremely impractical to resimulate the full event set each time the theory makes progress, or to simulate separate event sets using different parameter values in order to estimate theoretical uncertainties.

All of these problems can be solved if the exact distribution of the generated events is known. If this is the case, one can reweight the generated events according to

a different theoretical distribution, even after the time-consuming detector simulation. This makes it possible to reuse existing simulated events, and thus allows one with small effort to study theoretical uncertainties and to include theoretical improvements whenever they become available.

For this reweighting approach to be possible, one has to have control of the precise functional form of the probability distribution underlying the event generator. For the matrix element generator, this is always the case by construction. However, it also requires one to have analytic control of the parton shower, which is generally not the case for the currently used parton showers. The reason is that a realistic parton shower needs to enforce four-momentum conservation at each vertex. While this is strictly speaking a subleading effect, it typically generates cross correlations between different splittings. Hence, although the basic probability distribution governing a single splitting can be obtained analytically, the analytic control over the full distribution is lost in the standard parton shower algorithms because of the way four-momentum conservation is enforced.

The main purpose of this paper is to show how the analytic control over the full probability distribution can be regained, and that in this way the reweighting approach becomes feasible. In the next section we review the basic parton shower and set up our notation. In Sec. III we take the parton shower of `Sherpa` [8], which has the same algorithm as `Pythia`'s virtuality-ordered parton shower [9, 10, 11], as an example to investigate a real parton shower algorithm in detail. We then show in Sec. IV how it can be modified to satisfy four-momentum conservation at each vertex, while at the same time retaining the analytic control of the probability distribution. In Sec. V we discuss how these results can be applied in the different contexts described above, and Sec. VI contains our conclusions.

II. SETUP

A. Branching Probabilities

The purpose of this section is to review some of the basics of parton showers [12] needed in our discussion and, in the process, introduce our notation. To be specific, we consider a final-state parton shower with the invariant mass as evolution variable. For simplicity, we assume all particles to be massless, although particle masses can be included in a straightforward way.

The branching of a mother parton with some energy E into two daughter partons is determined by two independent kinematic variables, which are chosen to be the invariant mass of the mother t (or equivalently the total invariant mass of the daughters) and the energy splitting z, which determines how the mother's energy is distributed between the daughters. Before the mother is branched, it is still on shell with $t = 0$. During the branching step the mother is put off shell and given an invariant mass $t > 0$. At the same time the energy splitting z is obtained.

The single branch probability $\mathcal{P}(t, z)$, defined as the differential probability for a branching to occur with certain values t and z, is given by

$$\begin{aligned}\mathcal{P}(t,z) &= f(t,z)\exp\left\{-\int_t^{t_{\max}} \mathrm{d}t' \int_{\frac{1}{2}-z_{\mathrm{cut}}}^{\frac{1}{2}+z_{\mathrm{cut}}} \mathrm{d}z'\, f(t',z')\right\} \\ &\equiv f(t,z)\,\Pi(t,t_{\max})\,, \end{aligned} \tag{1}$$

where $f(t, z)$ is the usual Altarelli-Parisi splitting function [13] and $\Pi(t, t_{\max})$ is the well-known Sudakov factor [14], which resums the leading Sudakov double logarithms. The exact form of $t_{\max}$ and z_{cut} in Eq. (1) depends on the details of the parton shower implementation and will be discussed later.

The algorithm to determine the value of t at which the branching occurs is thought of as evolving t from some high start value $t_{\max}$ down to some lower value. The Sudakov factor $\Pi(t, t_{\max})$ then corresponds to the no-branching probability, i.e., the probability that no branching between $t_{\max}$ and t occurs. To see this, one integrates $\mathcal{P}(t, z)$,

$$\int_{t_1}^{t_{\max}} \mathrm{d}t \int_{\frac{1}{2}-z_{\mathrm{cut}}}^{\frac{1}{2}+z_{\mathrm{cut}}} \mathrm{d}z\, \mathcal{P}(t,z) = 1 - \Pi(t_1, t_{\max})\,. \tag{2}$$

The left-hand side is the probability for the mother to branch somewhere between t_1 and $t_{\max}$. Since the total probability is unity, $\Pi(t_1, t_{\max})$ is the probability that the mother does not branch between t_1 and $t_{\max}$.

When the mother is branched, the daughters are still on shell with zero invariant mass. In the next step the daughters themselves are branched and given non zero invariant masses, and after that, their daughters, and so on, resulting in a treelike structure as shown in Fig. 1. As the invariant mass of each daughter must be less than that of its mother, t is always decreasing for consecutive branchings. When a branching with $t < t_{\mathrm{cut}}$ is obtained, the corresponding mother is left unbranched. The parton shower terminates once no more branchings with $t \geq t_{\mathrm{cut}}$ are found. The value of the cutoff t_{cut} is typically chosen to be a low scale of order a few GeV^2.

Each event produced by the parton shower consists of a tree of n branches characterized by the set of variables $\{t_i, z_i\} \equiv \{t_1, z_1; t_2, z_2; \ldots; t_n, z_n\}$, which can later be turned into momentum four-vectors. As discussed in the Introduction, we would like to be able to explicitly compute the probability for a given event with certain values $\{t_i, z_i\}$ to be generated. Ideally, this probability is simply given as the product of individual single branch probabilities, schematically,

$$P_{\mathrm{PS}}(\{t_i, z_i\}) = \mathcal{P}(t_1, z_1) \times \mathcal{P}(t_2, z_2) \times \cdots\,. \tag{3}$$

If the parton shower satisfies Eq. (3), the problem of finding a closed form for $P_{\mathrm{PS}}(\{t_i, z_i\})$ reduces to working out the exact form of the single branch probability $\mathcal{P}(t, z)$.

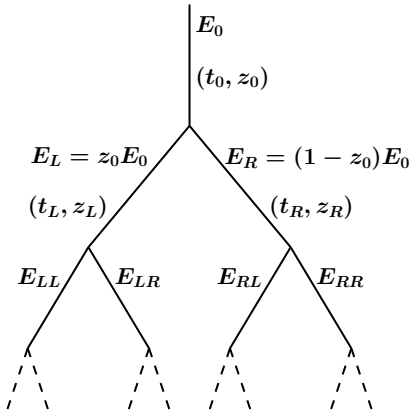

FIG. 1: Diagrammatic representation of a tree of branches.

It is important to point out that, even though the parton shower itself is only valid in the limit where the invariant mass of each daughter is much smaller than that of its mother, we still need to know the exact form of $P_{\rm PS}(\{t_i, z_i\})$ for all values of $\{t_i, z_i\}$. In other words, it is not sufficient for our purposes to only know $P_{\rm PS}(\{t_i, z_i\})$ expanded in the limit where the parton shower is valid.

Eq. (3) would be trivially satisfied if all branchings were independent of each other. In general, this is not the case for two reasons. First, the branching of each daughter depends on the initial conditions set by its mother, which implies that $P_{\rm PS}(\{t_i, z_i\})$ depends on the specific structure of the tree. However, as long as each branching only depends on previous branchings, the total probability can still be written as a product of single branch probabilities as in Eq. (3).

The second reason Eq. (3) can be violated is more involved and related to the basic issue of the parton shower we wish to address. At the time each branch is generated, the corresponding daughters have not yet been branched and are still on shell. However, the phase-space limits for the branch following from four-momentum conservation depend on the daughters' final invariant masses. Hence, only after both daughters have been branched themselves, one can come back and enforce the correct phase-space limits on the branch. Usually this step involves some kinematic reshuffling, which ends up introducing a complicated correlation between the daughters' branches and thereby violating Eq. (3).

B. Kinematics

We begin by working out the phase space for a single $1 \to 2$ branch. We denote all kinematic quantities (see Fig. 1) related to the mother particle with a subscript 0 and those of the left and right daughters with a subscript L and R, respectively. We regard the mother's invariant mass t_0 and energy E_0 as fixed and take the daughters to have invariant masses t_L and t_R. The energy splitting z_0 is regarded as a property of the mother (or rather of the whole branch) linking the daughters' energies to E_0,

$$E_L = z_0 E_0\,, \qquad E_R = (1 - z_0) E_0\,. \tag{4}$$

Thus, energy conservation is automatically satisfied, and E_L and E_R are not free variables.

In the rest frame of the mother, the value of z_0 is fixed in terms of t_L and t_R,

$$z_0^{\rm CM} = \frac{1}{2}\left(1 + \frac{t_L}{t_0} - \frac{t_R}{t_0}\right). \tag{5}$$

Using Eqs. (4) and (5), the magnitude of the daughters' three-momenta is

$$\mathbf{p}_{\rm CM}^2 = E_L^2 - t_L = E_R^2 - t_R = \frac{t_0}{4}\lambda(t_0, t_L, t_R)^2\,, \tag{6}$$

with the usual phase-space factor

$$\lambda(t_0, t_L, t_R) = \frac{1}{t_0}\sqrt{(t_0 - t_L - t_R)^2 - 4 t_L t_R}\,. \tag{7}$$

To enforce timelike daughters one needs

$$t_L \le E_L^2\,, \qquad t_R \le E_R^2\,, \tag{8}$$

which, using Eqs. (6) and (7), is equivalent to the usual phase-space limit

$$\sqrt{t_L} + \sqrt{t_R} \le \sqrt{t_0}\,. \tag{9}$$

The kinematics in a general frame is obtained by boosting the above results. The magnitude of the boost $\beta_0 = \sqrt{1 - t_0/E_0^2}$ is fixed by E_0 and t_0, while its direction can be described by the angle θ_0 between the boost axis and the daughters' three-momenta in the mother's rest frame. Since θ_0 encodes the information how the energies of the daughters are boosted relative to the mother's rest frame, it effectively determines z_0,

$$\begin{aligned} z_0 &= z_0^{\rm CM} + \beta_0 \cos\theta_0 \frac{|\mathbf{p}_{\rm CM}|}{\sqrt{t_0}} \\ &= \frac{1}{2}\left[1 + \frac{t_L}{t_0} - \frac{t_R}{t_0} + \beta_0 \cos\theta_0 \lambda(t_0, t_L, t_R)\right], \end{aligned} \tag{10}$$

and vice versa,

$$\cos\theta_0 = \frac{1}{\beta_0}\,\frac{2z_0 - (1 + t_L/t_0 - t_R/t_0)}{\lambda(t_0, t_L, t_R)}\,. \tag{11}$$

The phase-space limits in a general frame are a simple generalization of the limits in the rest frame,

$$\sqrt{t_L} + \sqrt{t_R} \le \sqrt{t_0}\,, \qquad |\cos\theta_0| \le 1\,. \tag{12}$$

Using Eq. (11), the limit on $\cos\theta_0$ is equivalent to the z_0 limit

$$\left| z_0 - \frac{1}{2}\left(1 + \frac{t_L}{t_0} - \frac{t_R}{t_0}\right)\right| \le \frac{\beta_0}{2}\lambda(t_0, t_L, t_R)\,, \tag{13}$$

commonly found in parton shower algorithms.

Finally, we look at a double branch $1 \to 2 \to 4$ where each daughter further branches into two on-shell particles. In this case, the daughters' energy splittings $z_{L,R}$, or equivalently $\theta_{L,R}$, are additional free variables. The complete phase space now is just an extension of Eq. (12),

$$\sqrt{t_L}+\sqrt{t_R} \le \sqrt{t_0}\,, \quad |\cos\theta_0| \le 1\,, \quad |\cos\theta_{L,R}| \le 1\,. \tag{14}$$

The limits on $z_{L,R}$ equivalent to $|\cos\theta_{L,R}| \le 1$ are analogous to Eq. (13) with the daughters' invariant masses set to zero. Hence, the complete phase space in terms of $t_{L,R}$ and $z_{0,L,R}$ reads

$$\begin{aligned} \sqrt{t_L} + \sqrt{t_R} &\le \sqrt{t_0}\,, \\ \left| z_0 - \frac{1}{2}\left(1 + \frac{t_L}{t_0} - \frac{t_R}{t_0}\right)\right| &\le \frac{\beta_0}{2}\lambda(t_0, t_L, t_R)\,, \\ \left| z_L - \frac{1}{2}\right| &\le \frac{\beta_L}{2}\,, \\ \left| z_R - \frac{1}{2}\right| &\le \frac{\beta_R}{2}\,, \end{aligned} \tag{15}$$

with

$$\beta_i = \sqrt{1 - t_i/E_i^2}\,, \tag{16}$$

and $E_{L,R}$ as in Eq. (4).

Eq. (15) explicitly shows the problem mentioned at the end of the previous section. Initially, z_0 is generated assuming $t_{L,R} = 0$, but since the limit on z_0 depends on t_L and t_R, the generated value of z_0 has to be adjusted after (t_L, z_L) and (t_R, z_R) have been determined. Changing z_0, however, changes $E_{L,R}$ and $\beta_{L,R}$. This in turn changes the limits on z_L and z_R, which can render their values invalid. In addition, t_L and t_R are determined independently from one another, so the constraint $\sqrt{t_L} + \sqrt{t_R} \le \sqrt{t_0}$ can be violated as well.

III. A STANDARD PARTON SHOWER

To study a concrete example of a standard parton shower, we consider the final-state parton shower of `Sherpa` [8], which employs the same algorithm as the `Pythia` virtuality-ordered parton shower [9, 10, 11]. Other algorithms which employ different ordering variables can be found in Refs. [15, 16, 17, 18].

A. The Algorithm

To be able to enforce four-momentum conservation, the parton shower algorithm always branches two sisters in pairs. That is, in each iteration it takes an existing $1 \to 2$ branch, consisting of a mother and two unbranched daughters, and converts it into a $1 \to 2 \to 4$ double branch by branching both daughters. To do so, the algorithm proceeds in three steps as depicted in Fig. 2:

1. Branch both daughters, each according to $\mathcal{P}(t, z)$.
2. Shuffle $z_0 \to z_0^{\text{new}}(z_0, t_L, t_R)$.
3. Check kinematics in terms of new z_0^{new}:
   (a) If successful, accept daughter branches.
   (b) If failed, evolve daughter with larger t further down and return to step 2.

In step 1, each daughter is branched separately, with values for (t_L, z_L) and (t_R, z_R) distributed according to the single branch probability $\mathcal{P}(t, z)$. In step 2, z_0 is changed to a new value $z_0^{\text{new}}(z_0, t_L, t_R)$, which is derived from its old value and takes into account the now nonzero values of $t_{L,R}$. In the mother's restframe this shuffling simply sets z_0 to the correct value z_0^{CM} in Eq. (5). In a general frame the form of $z_0^{\text{new}}(z_0, t_L, t_R)$ is not dictated by kinematics anymore, but is usually chosen to satisfy Eq. (13). In step 3, the kinematics are checked, using the new value z_0^{new}. If they are satisfied, the daughter branches are accepted. Otherwise, the algorithm takes the daughter with the larger t, evolves it further down, and goes back to step 2.

In the remainder of this section we discuss the details of this algorithm. We first work out the precise form of the single branch probability $\mathcal{P}(t, z)$ employed by the algorithm. We then move to discuss in detail steps 2 and 3, which implement four-momentum conservation, but as one can already see from Fig. 2 and the above discussion, introduce a complicated correlation between t_L and t_R, which clearly violates Eq. (3).

B. The Single Branch Probability

In this section we are only interested in a single $1 \to 2$ branching of a mother into two daughters. This means that each of the daughters in the algorithm described above acts as the mother now, and similarly, the mother and the other daughter in the algorithm now act as grandmother and sister, respectively.

In the first step, the algorithm independently generates two sets of values (t_L, z_L) and (t_R, z_R) according to the single branch probability $\mathcal{P}(t, z)$, introduced in Eq. (1). The precise form of $\mathcal{P}(t, z)$ that is actually used in the algorithm can be written as

$$\begin{aligned} \mathcal{P}(t, z) = f(t, z)\,\Pi(t, t_{\max})\,\theta(t_{\max} - t)\,\theta(t - t_{\text{cut}}) \\ \times\,\theta\Big[z_{\text{cut}}(t) - \Big|z - \frac{1}{2}\Big|\Big]\,, \end{aligned} \tag{17}$$

with the Sudakov factor

$$\Pi(t_1, t_2) = \exp\left\{-\int_{t_1}^{t_2} \mathrm{d}t \int_{\frac{1}{2}-z_{\text{cut}}(t)}^{\frac{1}{2}+z_{\text{cut}}(t)} \mathrm{d}z\, f(t, z)\right\}. \tag{18}$$

In Eq. (17) we explicitly included all kinematic θ functions restricting the allowed ranges of t and z. All information on the precise form of $\mathcal{P}(t, z)$ is encoded in

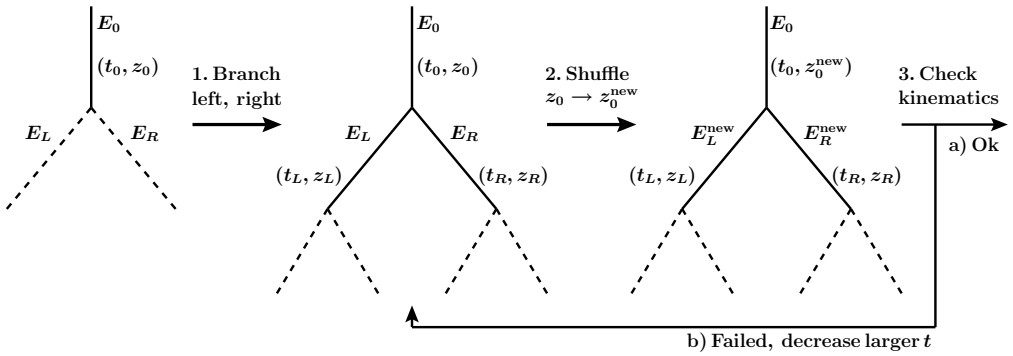

FIG. 2: Diagrammatic representation of a standard parton shower algorithm. Solid lines represent off-shell partons with nonzero invariant mass, dashed lines unbranched, on-shell partons.

the integration limits $t_{\max}$ and $z_{\text{cut}}(t)$, and the splitting function $f(t,z)$. Note that $\mathcal{P}(t,z)$ is not normalized to unity, because it describes the differential distribution in (t,z) for an entire $1 \to 2$ branch. It does not include the probability that the mother parton does not branch, in which case z is undefined. However, we can still define the differential distribution in t for a single parton,

$$\mathcal{P}(t) = \Pi(t_{\text{cut}}, t_{\max})\,\delta(t) + \int \mathrm{d}z\,\mathcal{P}(t,z)\,, \qquad (19)$$

where the first term is the no-branching probability, and $\mathcal{P}(t)$ is now properly normalized to unity,

$$\int \mathrm{d}t\,\mathcal{P}(t) = \Pi(t_{\text{cut}}, t_{\max}) + [1 - \Pi(t_{\text{cut}}, t_{\max})] = 1\,. \quad (20)$$

The precise form of the splitting function $f(t,z)$ depends on the specific type of splitting ($q \to qg$, $g \to qq$, or $g \to gg$), e.g.

$$f_{q\to qg}(t,z) = \frac{\alpha_s(\mu) C_F}{2\pi}\,\frac{1}{t}\,\frac{1+z^2}{1-z}\,. \qquad (21)$$

The scale at which α_s is evaluated is, in general, a function of t and z. For simplicity we use $\mu^2 = t/4$. Another typical choice is $\mu^2 = z(1-z)t$.

We stress that the kinematics relevant to Eq. (17) assumes that both daughters and the mother's sister are on-shell, massless particles. The expressions for $t_{\max}$ and $z_{\text{cut}}(t)$ arising from the phase space limits are then

$$t_{\max} = E_{\text{ini}}^2\,, \qquad z_{\text{cut}}(t) = \frac{\beta}{2}\,, \qquad (22)$$

where E_{ini} is the initial energy of the mother[1], and $\beta = \sqrt{1 - t/E_{\text{ini}}^2}$ in this subsection.

[1] Usually, E_{ini} is given in terms of the grandmother's z_0 and E_0. One exception is the case when the mother is the final parton coming from a hard interaction in the grandmother's rest frame. In this case, E_{ini} is chosen to be $E_{\text{ini}}(t) = \sqrt{t_0}(1 + t/t_0)/2$, which is the exact result for an on-shell, massless sister.

In addition to the pure phase space limits, the algorithm includes several restrictions which modify the form of $t_{\max}$ and $z_{\text{cut}}(t)$. First, since the parton showers is ordered in the evolution variable, t is always smaller than the initial value t_{ini} where the evolution starts, which is usually chosen to be the invariant mass of the grandmother, $t_{\text{ini}} = t_0$. Second, the cutoff on the algorithm, $t \geq t_{\text{cut}}$, is realized as a cut on $p_T^2 \equiv |\mathbf{p}_T|^2 \geq t_{\text{cut}}/4$, where $\mathbf{p}_T$ is the daughters' transverse momentum with respect to the mother's flight direction. In terms of t and z we have

$$p_T^2 = \frac{t}{4} - \frac{t}{\beta^2}\Big(z - \frac{1}{2}\Big)^2\,. \qquad (23)$$

Hence, $p_T^2 \geq t_{\text{cut}}/4$ implies $t \geq t_{\text{cut}}$, but with the additional restriction

$$\left|z - \frac{1}{2}\right| \leq \frac{\beta}{2}\sqrt{1 - \frac{t_{\text{cut}}}{t}}\,. \qquad (24)$$

Furthermore, parton showers require the opening angle between the daughters of subsequent emissions to always decrease. This angular ordering ensures that the branching is correctly described not only in the collinear limit, where the splitting functions are derived, but also in the soft limit, where the branching is coherent [9, 19]. The opening angle ϑ is given by

$$\cos\vartheta = 1 - \frac{t}{2z(1-z)E_{\text{ini}}^2}\,, \qquad (25)$$

which allows us to translate the angular ordering cut $\vartheta \leq \vartheta_{\text{cut}}$, where ϑ_{cut} is the opening angle of the previous branch, into limits on t and z,

$$t \leq E_{\text{ini}}^2 \frac{1 - \cos\vartheta_{\text{cut}}}{2}\,,$$
$$\left|z - \frac{1}{2}\right| \leq \frac{\beta}{2}\sqrt{1 - \frac{t}{\beta^2 E_{\text{ini}}^2}\,\frac{1+\cos\vartheta_{\text{cut}}}{1-\cos\vartheta_{\text{cut}}}}\,. \qquad (26)$$

Putting everything together, the single branch probability $\mathcal{P}(t,z)$ depends on several additional parameters

restricting the allowed range of t and z,

$$\mathcal{P}(t,z) \equiv \mathcal{P}(t,z|t_{\rm ini},E_{\rm ini},t_{\rm cut},\vartheta_{\rm cut})\,, \tag{27}$$

where the limits on t and z are determined by

$$t_{\rm max} = \min\left\{ t_{\rm ini}\,, E_{\rm ini}^2 \frac{2}{1-\cos\vartheta_{\rm cut}} \right\}, \tag{28}$$

$$z_{\rm cut}(t) = \frac{\beta}{2}\min\left\{ \sqrt{1-\frac{t}{\beta^2 E_i^2}\,\frac{1+\cos\vartheta_{\rm cut}}{1-\cos\vartheta_{\rm cut}}}\,, \sqrt{1-\frac{t_{\rm cut}}{t}} \right\}.$$

Note that these limits automatically include the phase-space limits, Eq. (22), which are reproduced for $\vartheta_{\rm cut}=\pi$ and $t_{\rm cut}=0$. Also, for $t\to E^2$, corresponding to the mother's rest frame, $z_{\rm cut}(t)\to 0$, forcing $z\to 1/2$, as required.

C. Enforcing Momentum Conservation

In each iteration the algorithm in Fig. 2 starts with a value for z_0 which, as discussed above, was determined in a previous iteration assuming $t_{L,R}=0$ and satisfying $|2z_0-1|\le\beta_0$. In particular, in the mother's rest frame it starts with $z_0=1/2$, whereas after step 1, t_L and t_R have become nonzero and so the correct value is now z_0^{CM} given in Eq. (5). Thus, the value of z_0 has to be changed (shuffled), or otherwise the kinematics can never be satisfied.

This shuffling happens in step 2 of the algorithm. There are various ways to do this, and `Sherpa` uses

$$\begin{aligned} z_0 \to z_0^{\rm new}(z_0,t_L,t_R) \\ = \frac{1}{2}\left[1+\frac{t_L}{t_0}-\frac{t_R}{t_0}+(2z_0-1)\lambda(t_0,t_L,t_R)\right], \end{aligned} \tag{29}$$

with $\lambda(t_0,t_L,t_R)$ given in Eq. (7). For $z_0=1/2$, Eq. (29) reduces to Eq. (5), as required. Since the original value of z_0 satisfies $|2z_0-1|\le\beta_0$, the shuffling in Eq. (29) ensures that z_0 always satisfies the correct phase-space limit, Eq. (13), for $t_{L,R}\neq 0$. Nevertheless, four-momentum conservation can still be violated in two ways. First, it may not be possible at all to find a new physical value $z_0^{\rm new}$. Namely, $t_{L,R}$ may not satisfy the constraint $\sqrt{t_L}+\sqrt{t_R}\le\sqrt{t_0}$ in Eq. (15), which ensures timelike daughters and that $\lambda(t_0,t_L,t_R)$ in Eq. (29) is well defined. This can happen, because the values of t_L and t_R were determined independently from one another, only subject to the constraint $t_{L,R}\le t_{\rm max}$ [see Eq. (28)].

Second, changing $z_0\to z_0^{\rm new}$ also changes the energies of the two daughters,

$$E_L\to E_L^{\rm new}=z_0^{\rm new}E_0\,,\quad E_R\to E_R^{\rm new}=(1-z_0^{\rm new})E_0\,, \tag{30}$$

and accordingly [see Eq. (16)],

$$\beta_{L,R}\to\beta_{L,R}^{\rm new}=\sqrt{1-t_{L,R}/E_{L,R}^{\rm new}}\,. \tag{31}$$

Now assume, for instance, that $t_L>t_R$. Then $z_0^{\rm new}>z_0$, which increases E_L (decreases E_R) and decreases β_L (increases β_R). Using Eq. (15), it follows that the available phases space for z_L shrinks, and hence the value of z_L may not be allowed anymore. The same is true for the right branch in case $t_R>t_L$.

For this reason, the algorithm has to explicitly check the kinematics, which is done in step 3. In `Sherpa` this is implemented by checking various different kinematical constraints arising from four-momentum conservation, all of which can be reduced to the constraints in Eq. (15). If all constraints are satisfied, the daughter branches are accepted and the algorithm proceeds with the next iteration. Otherwise, if any phase-space limit is violated, the algorithm picks the daughter with the larger t, generates new values (t,z) according to $\mathcal{P}(t,z)$ with the previous t as $t_{\rm ini}$, and goes back to step 2.

It should be clear from this discussion that steps 2 and 3 in the algorithm introduce a complicated cross correlation between the probabilities to get certain values (t_L,z_L) and (t_R,z_R), which breaks the factorization in Eq. (3), and makes it virtually impossible to find a closed-form expression for the double branch probability $P(t_L,z_L;t_R,z_R)$.

To end this section, we note that the shuffling of z_0 is proportional to $t_{L,R}/t_0$. As the parton shower is formally only valid for $t_{L,R}\ll t_0$, this is formally a power suppressed effect. Four-momentum conservation, however, is an important power correction that must be taken into account in order to obtain realistic events, also because in the end the parton shower is used for any values $t_{L,R}\le t_0$.

IV. THE ANALYTIC PARTON SHOWER

A. The Analytic Algorithm

It is the final step in the parton shower, where a generated set (t_L,z_L) or (t_R,z_R) can be rejected, which leads to the complicated correlation in the double branch probability. As explained above, there are two reasons the kinematics can fail in step 3. First, it is possible that the original values of $t_{L,R}$ violate $\sqrt{t_L}+\sqrt{t_R}\le\sqrt{t_0}$, and second, the required momentum shuffling in step 2 can render the values of $z_{L,R}$ invalid. We now show how both of these problems can be dealt with without introducing complicated correlations.

We start by looking at the second problem, for which it is instructive to understand in more detail the physical picture behind the shuffling in Eq. (29). Looking back at Eq. (10), one can think of z_0 in a general frame as given in terms of $t_{0,L,R}$ and $\beta_0\cos\theta_0$. Vice versa, for given $t_{0,L,R}$, z_0 determines the boost factor $\beta_0\cos\theta_0$. When z_0 was generated, $t_{L,R}=0$, and Eq. (10) implies

$$\beta_0\cos\theta_0 = 2z_0-1\,. \tag{32}$$

Thus, selecting a value for z_0 is equivalent to choosing

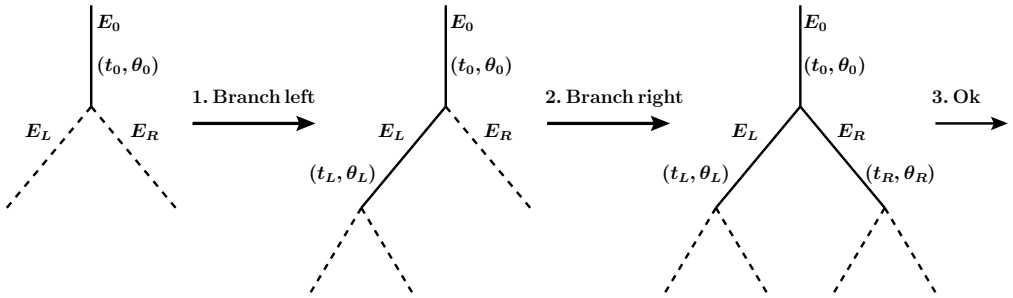

FIG. 3: Diagrammatic representation of the analytic algorithm. Solid lines represent off-shell partons with nonzero invariant mass, dashed lines unbranched, on-shell partons. The evolution of t_L in step 1 starts at $t_{\rm ini} = t_0$, and that of t_R in step 2 at $t_{\rm ini} = (\sqrt{t_0} - \sqrt{t_L})^2$.

the boost factor $\beta_0 \cos\theta_0$. With this in mind, it is easy to understand what the shuffling in Eq. (29) is doing physically. It simply holds $\beta_0 \cos\theta_0$ fixed at its generated value, and recomputes z_0 using Eq. (10) after the daughters acquire nonzero invariant masses. In other words, the algorithm treats the boost factor $\beta_0 \cos\theta_0$ as the fundamental and z_0 as the derived quantity.[2]

In the same way, generating values for $z_{L,R}$ in step 1 in Fig. 2 can be regarded as generating and fixing values for the boost factors

$$\beta_{L,R} \cos\theta_{L,R} = 2 z_{L,R} - 1 \,. \tag{33}$$

However, $\beta_{L,R}$ are not free quantities, but change with z_0 as functions of $t_{L,R}$ [see Eqs. (30) and (31)]. The only way to keep $\beta_{L,R} \cos\theta_{L,R}$ fixed when z_0 is shuffled is to balance the change in $\beta_{L,R}$ by a corresponding change in $\cos\theta_{L,R}$. In this picture, we now immediately see what can go wrong. When shuffling z_0, either β_L or β_R might decrease too much, resulting in an unphysical value $|\cos\theta| > 1$ for either θ_L or θ_R.

This picture also leads to a simple and general solution to the problem: Instead of $\beta \cos\theta$, we can just as well use the boost angle θ itself as the fundamental quantity. When a value for z is generated, it is translated into a value for $\cos\theta$,

$$\cos\theta = \frac{2z - 1}{\beta} \,, \tag{34}$$

which is held fixed at any later stage in the algorithm. The advantage is that the phase-space limits $|\cos\theta| \leq 1$ are completely independent of any other kinematic variables and are always satisfied. Holding $\cos\theta$ fixed in Eq. (34) corresponds to an additional shuffle

$$z \to z^{\rm new}(z, \beta, \beta^{\rm new}) = \frac{1}{2}\left[1 + (2z-1)\frac{\beta^{\rm new}}{\beta}\right], \tag{35}$$

to be applied to $z_{L,R}$ whenever $\beta_{L,R}$ change as a result of shuffling z_0. Similarly to Eq. (29), Eq. (35) ensures that $z_{L,R}$ always satisfy their phase-space constraints.

Note that Eqs. (29) and (35) can be realized as two special cases of a generalized shuffle, which follows from Eqs. (34) and (10),

$$\begin{aligned} z^{\rm old} \to z^{\rm new}(z^{\rm old}, \beta^{\rm old}, t_L, t_R, \beta^{\rm new}) \\ = \frac{1}{2}\left[1 + \frac{t_L}{t} - \frac{t_R}{t} + (2z^{\rm old} - 1)\frac{\beta^{\rm new}}{\beta^{\rm old}}\lambda(t, t_L, t_R)\right], \end{aligned} \tag{36}$$

where $\beta^{\rm old}$ is the value used when $z^{\rm old}$ was generated, and $\beta^{\rm new}$ is the new value. This makes the practical implementation into the current algorithms straightforward, because all one has to do is to store $\beta^{\rm old}$ and change the shuffling function.

Since all Eq. (36) does is to hold $\cos\theta$ fixed, we can also go one step further and directly use (t, θ) to describe the individual branches, with the daughters' energies $E_{L,R}$ always given as functions of E_0, θ_0 and $t_{0,L,R}$. In this way, any necessity to keep track of $z_{0,L,R}$ and how and when they are shuffled is removed, which makes the entire algorithm very transparent.

At this point, the only kinematical check that would be left in step 3 is the simple constraint $\sqrt{t_L} + \sqrt{t_R} \leq \sqrt{t_0}$, which cannot be eliminated by a change of variables. However, we can recast the correlation it introduces into a calculable form by branching the daughters in two separate steps, as shown in Fig. 3. In the first step the left daughter is branched, starting the evolution at $t_{\rm ini} = t_0$. In the second step the right daughter is branched, starting the evolution at $t_{\rm ini} = (\sqrt{t_0} - \sqrt{t_L})^2$, which automatically takes care of the remaining constraint. In this way, the double branch probability for the left and right branches can be written as the product of two single branch probabilities

$$\begin{aligned} P(t_L, \theta_L; t_R, \theta_R) = \mathcal{P}(t_L, \theta_L | t_{\rm ini} = t_0) \\ \times \mathcal{P}\big[t_R, \theta_R | t_{\rm ini} = (\sqrt{t_0} - \sqrt{t_L})^2\big] \,. \end{aligned} \tag{37}$$

Since the iteration steps in the generation of an entire event are already independent, Eq. (37) allows us to write

[2] This statement is true in general, even if z_0 is shuffled more than once, because in the algorithm the value of z_0 used on the right-hand side of Eq. (29) is always the originally generated value, never one which was already obtained from shuffling.

the total probability to generate a given event as a product of single branch probabilities as in Eq. (3), which is what we set out to do.

B. Practical Implementation

In practice, one has several choices in implementing this new analytic algorithm in a real parton shower. To eliminate the asymmetry between t_L and t_R in Eq. (37), one can randomly choose which daughter of a given branch acts as the left daughter and is branched first. Alternatively, one can always branch the daughter with the larger (or smaller) initial energy first. A third choice is to generate a test value of t for each daughter, call the larger one t_L, and then branch the other daughter starting the evolution at the smaller of t_L and $(\sqrt{t_0}-\sqrt{t_L})^2$. For this choice, the double branch probability still factorizes and involves an additional factor of the no-branching probability $\Pi(t_0, t_L)$. This last choice may be the most natural one from the point of view of a global evolution [20].

Once the left daughter has been branched, one also has the choice which energy to use as the initial energy E_{ini} for branching the right daughter. One could either keep the original energy computed from θ_0 with $t_L = 0$ or recompute it from θ_0 with the new value of t_L, where again the latter choice seems to be the more natural one.

We have implemented the changes to the algorithm described above in `Sherpa`'s parton shower. For simplicity, we always branch the left daughter first and keep the original energy for E_{ini} when branching the right daughter. The only things we had to change then were to separate the branching of the two daughters and to change the function returning a new value of z to use Eq. (36). As a cross check, we did not remove the kinematic checks done in step 3 of the original algorithm and tested that with our modifications they indeed never fail (apart from extremely rare occasions where the failure is due to numerical inaccuracies).

C. Comparison and Numerical Results

We now discuss the impact the change in the algorithm has on the generated events. The original algorithm rejects branches in the third step if the kinematics is not satisfied, which leads to a lowering of at least one of the invariant masses of the daughters that are branched. Hence, compared to the analytic algorithm, where this third step is absent, the original algorithm suppresses large values of t and enhances low values of t. Since the effect on the kinematics from having nonzero t is more pronounced for larger t, this relative suppression is expected to increase with increasing t.

To estimate the expected size of this effect, we note that shuffling z is a power suppressed effect, of order t/t_0, which one can think of as changing the z limits of integration. Thus, upon integration over z, the difference in the two algorithms corresponds to a power correction to the splitting function $f(t) \equiv \int \mathrm{d}z\, f(t,z)$, schematically

$$\Delta f(t) = f(t) \times \mathcal{O}\Big(\frac{t}{t_0}\Big)\,. \tag{38}$$

Furthermore, the single branch probability generated by the algorithm always has the form $P(t) = f(t)\,\exp[\int \mathrm{d}t\, f(t)]$. The integral of a difference in $f(t)$ in the exponent gives rise to an additional finite perturbative difference. Thus, Eq. (38) translates into a change in the single branch probability

$$\Delta P(t) = P(t) \times \left[\mathcal{O}\Big(\frac{t}{t_0}\Big) + \mathcal{O}(\alpha_s)\right]. \tag{39}$$

The appearance of perturbative corrections can also be understood in another way. Since the total probability $P(t)$ is normalized to unity [see Eq. (19)], and power corrections must vanish for $t \to 0$, an increase of $P(t)$ at large t via power corrections can only be compensated at small t by a decrease of $P(t)$ via perturbative corrections.

To illustrate the difference between the algorithms at the level of a single branching, we generated one million double branches starting from $t_0 = (91.2\,\text{GeV})^2$ in the mother's rest frame, and look at the average of the double branch probability

$$\begin{aligned}\bar{P}(t) = \int \mathrm{d}t_L\, \mathrm{d}z_L\, \mathrm{d}t_R\, \mathrm{d}z_R\, \frac{\delta(t-t_L)+\delta(t-t_L)}{2} \\ \times P(t_L, z_L; t_R, z_R)\,.\end{aligned} \tag{40}$$

The results are shown on the left of Fig. 4 for the original `Sherpa` algorithm [dark (blue) triangles] and the analytic algorithm [medium (orange) dots]. The solid (orange) line shows the analytic result for $\bar{P}(t)$ obtained from Eq. (37). As expected, it matches the distribution generated with the analytic algorithm. One can also see the suppression of the original algorithm at large t. The difference between the two algorithms for large t is well within the expected size of power corrections Eq. (39), indicated by the gray shading.

To show the effect on fully showered events, we generated one million events with `Sherpa` using the original and the analytic algorithm. A typical physical observable is the thrust distribution, $\mathrm{d}\sigma/\mathrm{d}T$, which measures the jettiness of a given event, with $T \to 1$ for two narrow jets and $T \to 1/2$ for spherical events. On the right of Fig. 4 we show the integrated thrust distribution

$$\hat{\sigma}(\tau) \equiv \frac{1}{\sigma}\int_{1-\tau}^{1} \mathrm{d}T\, \frac{\mathrm{d}\sigma}{\mathrm{d}T} \tag{41}$$

obtained with the original algorithm [dark (blue) triangles] and the analytic algorithm [medium (orange) dots]. Here, the gray shading gives an estimate of the size of perturbative corrections. As expected, compared to the original algorithm, the analytic algorithm suppresses small values of τ, corresponding to branchings at small t, but within the estimated size of perturbative corrections.

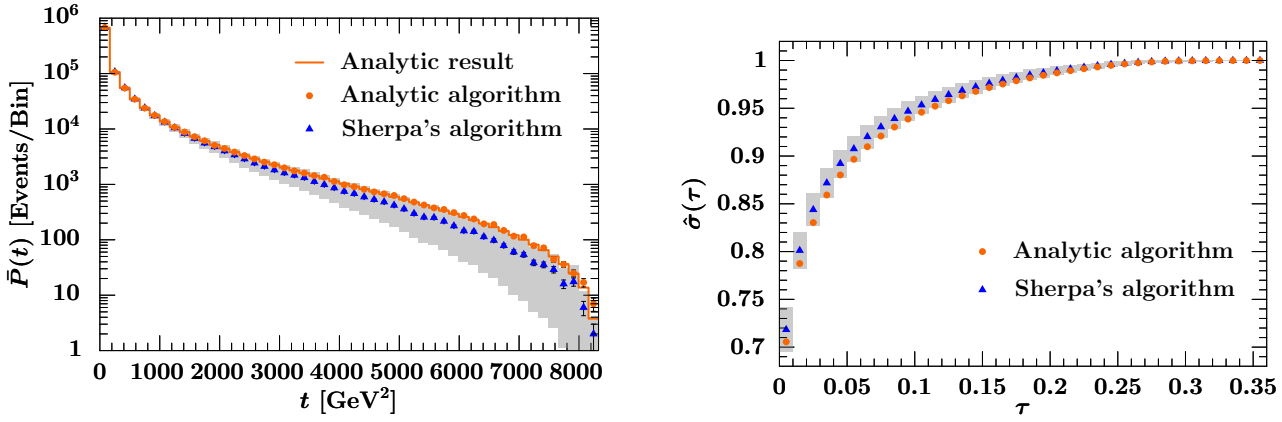

FIG. 4: Comparison of the analytic parton shower [medium (orange) dots] and Sherpa's parton shower [dark (blue) triangles]. Left: The function $\bar{P}(t)$ as defined in Eq. (40). The solid (orange) line shows the result obtained from Eq. (37), and the gray shading gives an estimate of the expected size of power corrections. Right: The integrated thrust distribution, where the gray shading shows the expected size of finite perturbative corrections. The error bars show the statistical uncertainties.

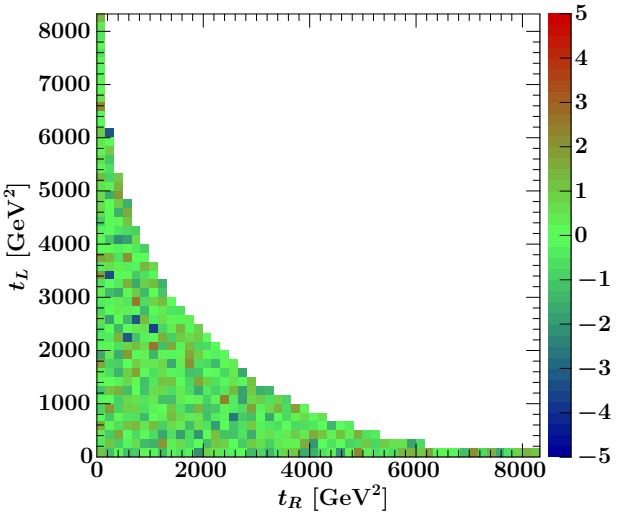

FIG. 5: Pull distribution for the double branch probability $P(t_L, t_R)$ defined in Eq. (42). See text for further explanation.

As a further check that the analytic algorithm distributes according to the known probability Eq. (37), we keep the left and right branches separate and consider the double branch probability integrated over $z_{L,R}$

$$P(t_L, t_R) = \int dz_L\, dz_R\, P(t_L, z_L; t_R, z_R)\,. \qquad (42)$$

The result of the analytic algorithm agrees well within the statistical uncertainties with the analytic expectation from Eq. (37). This is illustrated in Fig. 5, where we show the pull distribution for $P(t_L, t_R)$, i.e. the difference between the generated and analytic distributions divided by the statistical uncertainty of the generated distribution.

V. REWEIGHTING EVENTS

We have shown how to construct an analytic parton shower algorithm which distributes events according to a known probability distribution $P_{\rm PS}(\{t_i, z_i\})$. This result opens up the possibility to obtain events that are distributed according to some other distribution $P_{\rm new}(\{t_i, z_i\})$ by proceeding in three steps:

1. Generate events using the analytic algorithm described in this work.

2. Assign the weight $P_{\rm new}(\{t_i, z_i\})/P_{\rm PS}(\{t_i, z_i\})$ to each event.

3. If desired, unweight the event sample by vetoing events according to their relative weights.

The third step is optional and only needed if one desires final events with unit weight. Similarly, the allowable size of the weights will depend on the specific application. The power of this approach is twofold. First, it can provide a very efficient way to distribute according to some distribution $P_{\rm new}$, which may not be possible otherwise. Second, since the reweighting does not change the event kinematics, it can be applied at any later stage in the event generation, in particular, after the detector simulation. We like to stress again that, for this reweighting approach to be possible, it is essential to know the exact form of $P_{\rm PS}$, i.e., it would be insufficient to only know the leading terms in the power expansion of $P_{\rm PS}$. We now discuss two immediate applications.

A. Distributing and Matching Matrix Elements

We first consider the case where $P_{\rm new} = P_{\rm ME}$ is given by the differential distribution obtained from full matrix

element calculations. Even though it is possible to obtain the squared amplitude, and thus $P_{\rm ME}$, for a moderate number of partons in the final state, it is still quite difficult to distribute events according to $P_{\rm ME}$. The reason is that $P_{\rm ME}$ has large peaks arising from poles in the amplitudes due to on-shell intermediate particles. This makes it impossible to employ a simple hit-or-miss algorithm with random numbers drawn from a flat distribution. Instead one has to rely on numerically inverting the integral of $P_{\rm ME}$ over phase space. Since the dimensionality of phase space increases with each additional parton, the required numerical integrations quickly become very time-consuming.

The parton shower describes the same IR physics as the full QCD amplitude. It thus contains the same poles from on-shell intermediate particles, and the resulting weights $P_{\rm ME}/P_{\rm PS}$ in step 2 are expected to be moderate. Generating events with a parton shower is several orders of magnitude faster than the numerical phase-space integrations required otherwise. Hence, it should be possible to accommodate relative weights even of $\mathcal{O}(100)$ to $\mathcal{O}(1000)$ and still achieve a reasonably high efficiency. As the events will later be run through a detector simulation, step 3 has to be included such that the final events have unit weight. One way to think of this is that the parton shower acts as a phase-space generator that automatically contains the correct pole structure of the matrix elements.

From this viewpoint, one could consider a much simpler version of the Markov Chain algorithm that only attempts to capture the underlying pole structure, without trying to address other important effects (e.g. getting the correct soft limit via the angular ordering) that are already included in the matrix elements. This would provide an alternative and efficient algorithm to distribute events according to known matrix element expressions.

Nevertheless, to obtain realistic, fully exclusive events one still has to attach a parton shower to the matrix element calculations, which is usually nontrivial due to double counting issues. There are a few dedicated algorithms [4, 5, 6] with several implementations [21, 22, 23] available to consistently match tree-level matrix elements for many partons with parton showers. In addition, several approaches are pursued by now to incorporate matrix elements for the first hard emission at next-to-leading order [24, 25, 26, 27]. To combine these two separate classes of matrix element corrections, some work has been carried out in Ref. [28]. In this respect, the SCET-based approach of Ref. [2] seems quite promising.

Using the analytic parton shower, one can generate fully showered events with many partons in the final state and then reweight the n hardest emissions to the matrix element result for n final-state partons. This idea is similar to the older merging method used by `Pythia` to correct the first shower emission [9, 10, 11, 29]. In our case, one assigns to each event the weight $P_{\rm ME}(\{t_k, z_k\})/P_{\rm PS}(\{t_k, z_k\})$ in step 2, where $k = 1, \ldots, n$ numbers the n partons with the largest t. It follows that any observable sensitive to the distribution in the n hardest partons and inclusive in all other partons will be determined by the full matrix element result, while any further emission is determined by the parton shower. In this way the analytic parton shower not only allows one to efficiently distribute matrix elements, but in addition provides a simple and powerful tool to match matrix elements and parton showers. An implementation of this result will be given elsewhere. Note that the matching is completely determined at the analytic level by the form of $P_{\rm ME}$. In principle, it could be carried out for any n and at any order in perturbation theory.

B. Parton Shower Tuning and Uncertainties

The reweighting can also be performed after the events have been run through a detector simulation, which is the most time-consuming part of the event generation. Hence, one can obtain sets of events with different underlying distributions, performing only a single run of the detector simulation. Of course, to make maximal use of the simulated events, one would now like to have $\mathcal{O}(1)$ weights and also skip step 3. Therefore, to generate events one would still use the best available matrix element calculations matched with the analytic parton shower, as described above.

One advantage of using the analytic parton shower is that one can update already simulated events at any later time to the newest theory or parameters. Furthermore, having analytic control over all parameters in the parton shower allows one to estimate uncertainties arising from input parameters like $\alpha_s(m_Z)$ or quark masses, and from higher-order power and perturbative corrections. In all cases $P_{\rm new}$ is simply given by $P_{\rm PS}$ computed with a different set of parameters. Moreover, one could study different scheme choices in the parton shower, provided of course one has analytic control over $P_{\rm PS}$ corresponding to the new scheme. For example, one could study the scale at which α_s is evaluated or even the choice of the evolution variable.

As a specific application, we look at power corrections and the tuning of the parton shower. Tuning a parton shower is crucial to get a good description of the experimental data, and one is, in effect, adjusting unknown power corrections to fit the data. The analytic control over the parton shower gives access to a simple and systematic way to tune it to data. We can introduce power corrections by adding nonsingular terms to the splitting function $f(t, z)$ entering Eq. (17). For example, for $f_{q\to qg}(t, z)$ as in Eq. (21),

$$f_{\rm new}(t, z) = \frac{\alpha_s C_F}{2\pi}\left[\frac{1}{t}\frac{1+z^2}{1-z} + g(t)\right], \qquad (43)$$

where $g(t)$ is a nonsingular function of t (in general, g could also depend on z). This change will affect the branching probability for large t, but will leave branches

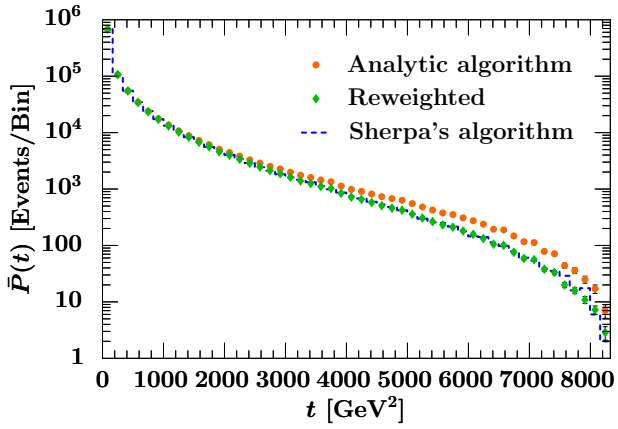

FIG. 6: The function $\bar{P}(t)$, as defined in Eq. (40), for the analytic parton shower [medium (orange) dots], after reweighting [light (green) diamonds], and for `Sherpa`'s parton shower [dashed (blue) line]. Error bars indicate statistical uncertainties. See text for further explanation.

with small t nearly unaffected. Hence, adjusting $g(t)$ changes the power corrections included in the parton shower, which can be used to tune the parton shower.

To illustrate this, we would like to adjust the analytic algorithm such that the average of its double branch probability, $\bar{P}(t)$ [see Eq. (40)], roughly agrees with the original algorithm. We use the simplest possible power correction, $g(t) = a/t_{\max}$. As discussed in Sec. IV C, the analytic algorithm enhances large t compared to the original algorithm, so we need $a < 0$. In Fig. 6 we show the result for $\bar{P}(t)$ from running the analytic algorithm [medium (orange) dots] and then reweighting each event to a new splitting function with $a = -1.5$ [light (green) diamonds], together with the result from running the original `Sherpa` algorithm [dashed (blue) line]. The reweighted result agrees remarkably well with the original algorithm, given the simple form of the power correction.

While this example only serves as an illustration, the analytic control over the tuning parameters is extremely useful. After the detector simulation, tuning the parton shower to data amounts to parametrizing the power corrections $g(t)$ in terms of a few parameters and fitting them to data. The uncertainties from power corrections can then be estimated by varying the tuning parameters in some range.

VI. CONCLUSIONS

Event generators are indispensable tools to compare theory and experiment at collider experiments. While distributions with relatively few final-state partons can be computed directly from the matrix elements of the underlying theory, parton showers have to be employed to generate final states with a large number of partons. They rely on splitting functions, which are derived in the collinear or soft limit of the underlying theory. The conservation of four-momentum in each step of the algorithm, although technically a subleading effect, is an important ingredient to obtain realistic predictions from parton showers to compare to data.

In standard parton shower algorithms, the implementation of four-momentum conservation introduces a cross correlation between different branches, such that the final probability to produce a given event is not equal to the product of individual branching probabilities. In this work we have shown that a few simple modifications of an existing algorithm yield an analytic parton shower algorithm that conserves four-momentum at each vertex, but distributes according to a known analytic distribution. This makes it possible to generate events with exact knowledge of the probability with which each event was generated, and to reweight the events to a different distribution at any later stage in the event generation.

We have studied the specific case of a virtuality-ordered final-state parton shower. Considering the type of modifications, we expect the extension to a corresponding initial-state parton shower to be straightforward. It should also be possible to apply the same ideas to parton showers using different ordering variables.

The analytic parton shower proposed here in conjunction with the reweighting approach provides a powerful tool for experiment and theory, and in the last section we have given two examples of this. First, it facilitates the distribution of events according to full matrix elements, which is otherwise hindered by the need for time-consuming phase-space integrations, and at the same time provides a very generic way to match matrix elements with the parton shower. Second, generated events can be reweighted even after they have been run through a detector simulation. This allows one to update simulated events at any later time. It also greatly simplifies the study of higher-order corrections and parameter dependences in the parton shower as well as the estimation of uncertainties arising from these sources. For example, it provides a convenient way to tune the parton shower to data by simply fitting parameters characterizing power corrections to the data.

Acknowledgments

We are grateful to Johan Alwall, Peter Skands, and Kerstin Tackmann for helpful conversations. This work was supported in part by the Director, Office of Science, Office of High Energy Physics of the U.S. Department of Energy under the Contract DE-AC02-05CH11231. CWB would also like to acknowledge support from the DOE OJI program, and an LDRD from LBNL.

[1] For a short review see: P. Z. Skands, AIP Conf. Proc. **792**, 73 (2005) [arXiv:hep-ph/0507129].

[2] C. W. Bauer and M. D. Schwartz, Phys. Rev. Lett. **97**, 142001 (2006) [arXiv:hep-ph/0604065]; arXiv:hep-ph/0607296;

[3] C. W. Bauer, S. Fleming and M. E. Luke, Phys. Rev. D **63**, 014006 (2000) [arXiv:hep-ph/0005275]; C. W. Bauer, S. Fleming, D. Pirjol and I. W. Stewart, Phys. Rev. D **63**, 114020 (2001) [arXiv:hep-ph/0011336]; C. W. Bauer and I. W. Stewart, Phys. Lett. B **516**, 134 (2001) [arXiv:hep-ph/0107001]; C. W. Bauer, D. Pirjol and I. W. Stewart, Phys. Rev. D **65**, 054022 (2002) [arXiv:hep-ph/0109045].

[4] M. L. Mangano, M. Moretti and R. Pittau, Nucl. Phys. **B632**, 343 (2002) [arXiv:hep-ph/0108069]; M. L. Mangano, Talk at the "Fermilab ME/MC Tuning Workshop", Oct. 4, 2002, http://home.fnal.gov∼mrenna/tuning/nov2002/mlm.pdf

[5] S. Catani, F. Krauss, R. Kuhn and B. R. Webber, JHEP **0111**, 063 (2001) [arXiv:hep-ph/0109231]; F. Krauss, JHEP **0208**, 015 (2002) [arXiv:hep-ph/0205283].

[6] L. Lönnblad, JHEP **0205**, 046 (2002) [arXiv:hep-ph/0112284].

[7] For other recent work on estimating uncertainties in parton showers see: P. Stephens and A. van Hameren, arXiv:hep-ph/0703240.

[8] R. Kuhn, F. Krauss, B. Ivanyi and G. Soff, Comput. Phys. Commun. **134**, 223 (2001) [arXiv:hep-ph/0004270]; F. Krauss, A. Schälicke and G. Soff, Comput. Phys. Commun. **174**, 876 (2006) [arXiv:hep-ph/0503087].

[9] M. Bengtsson and T. Sjöstrand, Phys. Lett. B **185**, 435 (1987); M. Bengtsson and T. Sjöstrand, Nucl. Phys. **B289**, 810 (1987).

[10] E. Norrbin and T. Sjöstrand, Nucl. Phys. **B603**, 297 (2001) [arXiv:hep-ph/0010012].

[11] T. Sjöstrand *et al.*, Comput. Phys. Commun. **135**, 238 (2001) [arXiv:hep-ph/0010017]; T. Sjöstrand, S. Mrenna and P. Skands, JHEP **0605**, 026 (2006) [arXiv:hep-ph/0603175].

[12] For a pedagogical introduction see: T. Sjöstrand, arXiv:hep-ph/0611247; and for a more detailed review the `Pythia` manual [11].

[13] G. Altarelli and G. Parisi, Nucl. Phys. **B126**, 298 (1977).

[14] V. V. Sudakov, Sov. Phys. JETP **3**, 65 (1956) [Zh. Eksp. Teor. Fiz. **30**, 87 (1956)].

[15] G. Marchesini and B. R. Webber, Nucl. Phys. **B238**, 1 (1984); G. Corcella *et al.*, JHEP **0101**, 010 (2001) [arXiv:hep-ph/0011363].

[16] G. Gustafson and U. Pettersson, Nucl. Phys. **B306**, 746 (1988); L. Lönnblad, Comput. Phys. Commun. **71**, 15 (1992).

[17] S. Gieseke, P. Stephens and B. R. Webber, JHEP **0312**, 045 (2003) [arXiv:hep-ph/0310083]; S. Gieseke *et al.*, JHEP **0402**, 005 (2004) [arXiv:hep-ph/0311208]; S. Gieseke *et al.*, arXiv:hep-ph/0609306.

[18] T. Sjöstrand and P. Z. Skands, Eur. Phys. J. C **39**, 129 (2005) [arXiv:hep-ph/0408302].

[19] A. H. Mueller, Phys. Lett. B **104**, 161 (1981).

[20] We thank Peter Skands for discussions on this point.

[21] T. Gleisberg *et al.*, JHEP **0402**, 056 (2004) [arXiv:hep-ph/0311263]; A. Schälicke and F. Krauss, JHEP **0507**, 018 (2005) [arXiv:hep-ph/0503281].

[22] S. Mrenna and P. Richardson, JHEP **0405**, 040 (2004) [arXiv:hep-ph/0312274].

[23] N. Lavesson and L. Lönnblad, JHEP **0507**, 054 (2005) [arXiv:hep-ph/0503293].

[24] S. Frixione and B. R. Webber, JHEP **0206**, 029 (2002) [arXiv:hep-ph/0204244]; S. Frixione, P. Nason and B. R. Webber, JHEP **0308**, 007 (2003) [arXiv:hep-ph/0305252]; S. Frixione and B. R. Webber, arXiv:hep-ph/0612272.

[25] M. Krämer and D. E. Soper, Phys. Rev. D **69**, 054019 (2004) [arXiv:hep-ph/0306222]; D. E. Soper, Phys. Rev. D **69**, 054020 (2004) [arXiv:hep-ph/0306268]; M. Krämer, S. Mrenna and D. E. Soper, Phys. Rev. D **73**, 014022 (2006) [arXiv:hep-ph/0509127].

[26] P. Nason, JHEP **0411**, 040 (2004) [arXiv:hep-ph/0409146]; P. Nason and G. Ridolfi, JHEP **0608**, 077 (2006) [arXiv:hep-ph/0606275].

[27] O. Latunde-Dada, S. Gieseke and B. R. Webber, JHEP **0702**, 051 (2007) [arXiv:hep-ph/0612281].

[28] Z. Nagy and D. E. Soper, JHEP **0510**, 024 (2005) [arXiv:hep-ph/0503053].

[29] G. Miu, arXiv:hep-ph/9804317; G. Miu and T. Sjöstrand, Phys. Lett. B **449**, 313 (1999) [arXiv:hep-ph/9812455].